

Though now, inconsistently with the Office's prior realization, the Office decided to revert back to the § 102(e) rejection of Claims 1-38 and 50. The reversion back to § 102(e) rejection by the Office appears to be inexplicable. Nevertheless, in this RESPONSE, the applicants are making another good faith effort to explain again why the § 102(e) rejection of all Claims 1-50 cannot be sustained.

CLAIM REJECTIONS — 35 U.S.C. § 102(e)

Claims 1-50 were rejected under 35 U.S.C. § 102(e) as being allegedly anticipated by Fernandez et al. U.S. Patent 6,785,673 B1 (hereafter "Fernandez"). (Office Action, page 3) This rejection is respectfully traversed.

CLAIM 1

CLAIM 1 recites:

1. A method comprising the computer-implemented steps of:
detecting that a portion of a query execution plan to service a request for data will cause a first producer execution unit that will perform said portion, according to said query execution plan, to generate XML data for use by a second consumer execution unit in performing another portion of said query execution plan;
generating information to send to said first execution unit to cause said first execution unit to perform said portion of said query execution plan;
wherein said information would cause said first execution unit to generate said XML data in a first form that cannot be used by said second execution unit; and
annotating said information with an annotation that causes XML data generated by said first execution unit to be transformed to a canonical form for use by said second execution unit in performing said another portion of said query execution plan,
wherein said annotating causes removal of one or more references to execution unit-specific data that is accessible by the first execution unit but that is not accessible by the second execution unit;
wherein the step of annotating includes annotating said information with an operator to transform said XML data to a canonical form in which said XML data is serialized to represent particular data for a particular XML construct and is included in a serialized image that is sent to said second execution unit;
wherein the step of annotating includes annotating said information with an operator to transform said XML data to a canonical form which includes an identifier of memory space where

data is persistently stored, and wherein said data in said memory space is accessible by said second execution unit.

To facilitate an understanding of Claim 1, a FIGURE is provided herein to illustrate the various features required to Claim 1. Specifically, Claim 1 recites:

“**detecting** that a portion of a query execution plan... will cause a first producer execution unit, that will perform said portion, to generate XML data for use by a second consumer execution unit in performing another portion of said query execution plan.”

In the FIGURE below, this feature is indicated by element 10.

Claim 1 further recites:

“**generating** information to send to said first execution unit to cause said first execution unit to perform said portion..., wherein said information would cause said first execution unit to generate said XML data in a first form that cannot be used by said second execution unit.”

In the FIGURE below, this feature is indicated by element 20.

Furthermore, Claim 1 recites:

“**annotating** said information with an annotation that causes XML data generated by said first execution unit to be transformed to a canonical form for use by said second execution unit in performing said another portion of said query execution plan.”

In the FIGURE below, this feature is indicated by element 30.

Moreover, Claim 1 also recites:

“said **annotating** causes removal of one or more references to execution unit-specific data that is accessible by the first execution unit but that is not accessible by the second execution unit.”

In the FIGURE below, this feature is indicated by element 40.

The elements 10, 20, 30 and 40 are depicted in FIGURE below on the left side of the FIGURE.

The remaining details of FIGURE are recited in the claims that dependent from Claim 1.

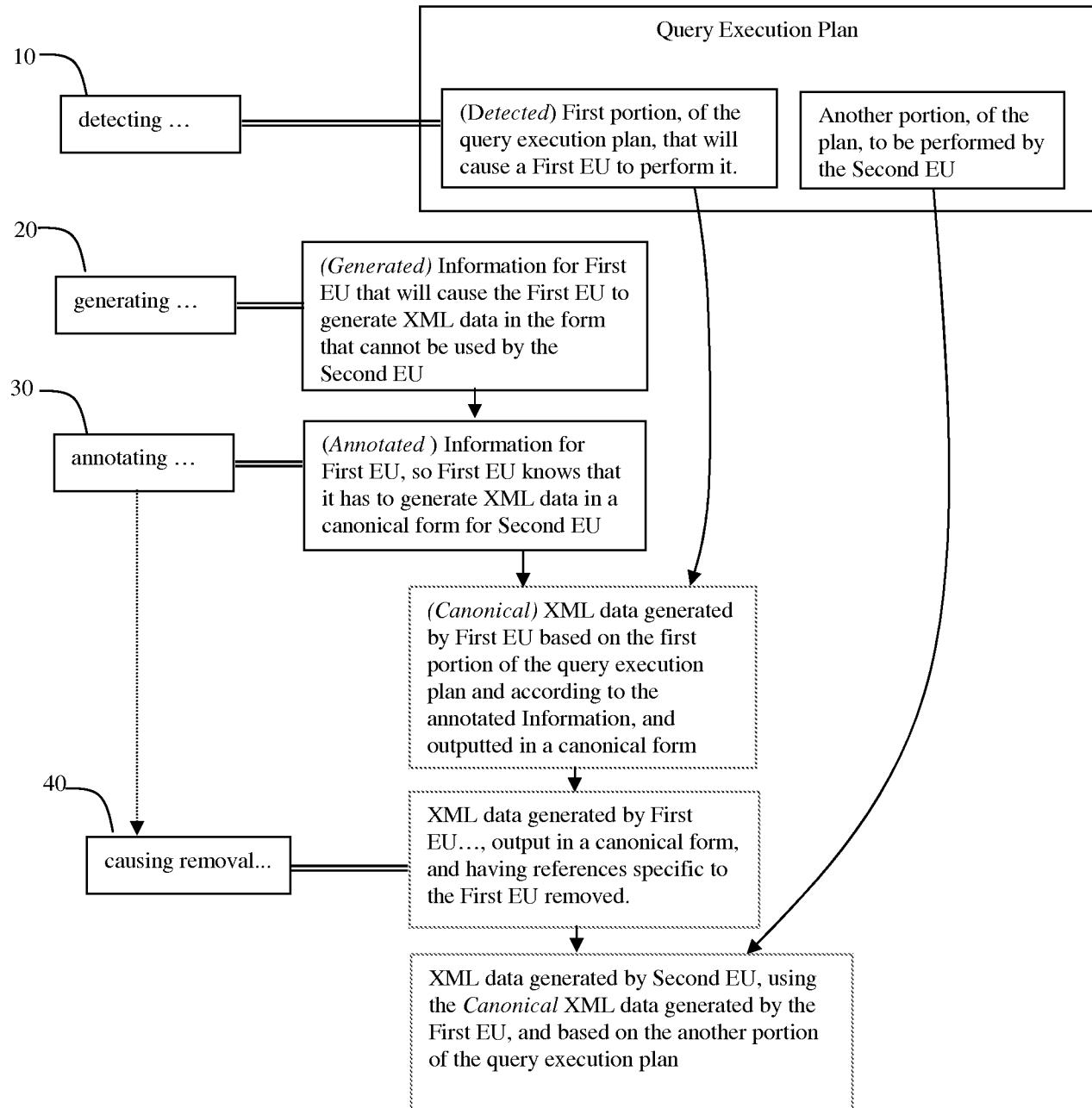


FIGURE: A block diagram illustrating a flow of information according to Claim 1.

As mentioned above, Claim 1 recites, and the FIGURE depicts, how the First Execution Unit processes the first portion of the execution query plan and generates XML data, in what format the XML data generated by the First Execution Unit is sent to the Second Execution Unit

and that the XML data generated by the First Execution Unit has references specific to the First Execution Unit removed.

The Office Action alleges that, in FIGS. 6-7, Fernandez anticipates

“detecting that a portion of a query execution plan... will cause a first producer execution unit, that will perform said portion, **to generate** XML data for use by a second consumer execution unit in performing another portion of said query execution plan.”.

This is incorrect.

FERNANDEZ FIG. 6

In Fig. 6, Fernandez depicts that a translator decomposes an RXL query into a set of SQL queries (Fernandez: Col. 32, ll. 56-57), and submits the SQL queries to a Relational database (RDBMS) (Fernandez: Col. 38, ll. 17-19). The RDBMS executes each of the SQL queries and returns results (Fernandez: col. 38, ll. 19-20), which are merged and integrated with logical relations into an output XML document sent to a user. However, Fernandez’ FIG. 6 does not show anything that would satisfy the express limitations of:

“detecting that a portion of a query execution plan... will cause a first producer execution unit, that will perform said portion, **to generate** XML data for use by a second consumer execution unit in performing another portion of said query execution plan.”.

For example, none of Fernandez’ RDBMS execution units “generate XML data for use by a second execution unit in performing another portion of said query execution plan,” as claimed.

FERNANDEZ FIG. 7

In FIG. 7, Fernandez depicts relations identified in the user’s query and RXL view query, used to generate SQL execution plans, and then utilized in producing an output XML document. More specifically, FIG. 7 of Fernandez depicts how the relations between SQL constructs for all

execution plans to be executed by the RDBMS are identified and stored, and later used to convert the SQL results into an output XML document. However, the relations themselves are not executed by the RDBMS. Fernandez' relations capture the nesting relations between the constructs in the SQL query execution plan, but are not "portions of a query execution plan," as claimed. Moreover, the relations are not "XML data, generated by the first execution unit, to be used by a second execution unit in performing another portion of the execution query," as claimed.

Fernandez uses the "relations," collectively called an XML template, to merge SQL query results and restore logical relations necessary to generate an answer in the form of an output XML document, but none of Fernandez' "relations" causes "the first execution unit to generate XML data for use by a second execution unit in performing another portion of the query execution plan," as claimed.

FERNANDEZ' XML TEMPLATE

Fernandez' XML template is used to convert the SQL query results to an output XML document, so the results are presented to the user in a form that is "compatible" with user's input XML query. The XML template preserves the nesting relations of structural constructs. Fernandez' column 38, line 24 to column 40, line 23 describes that, after the user enters a query in an XML format, the XML query is translated into a set of SQL queries, and Fernandez creates a View Tree to define a mapping between the XML query entities, relational constructs in the corresponding SQL queries and nested relationship of the constructs. Once the RDMBS returns SQL results, Fernandez uses the View Tree to integrate the SQL results (tuples) into an XML output document according to the saved mapping. To do the integration, Fernandez uses a well known Thoralf Skolem function, called the "Skolem function," which maps elements from one

set to elements from another set. In Fernandez, the Skolem function defines a mapping between RXL attributes and SQL constructs and helps to combine the SQL results into one XML output document. (Fernandez: col. 35, ll. 42-46) This is also depicted in FIG. 1, where the SQL queries results (Tuple Stream) are mapped onto an XML Template in order to generate an XML answer to the User XML Query. However, Fernandez' XML template is not “a portion of a query execution plan [...] executed by an execution unit,” as claimed. Fernandez’ XML template or a View Tree is not “XML data, generated by the first execution unit, to be used by a second execution unit in performing another portion of the execution query,” as claimed.

FERNANDEZ’ VIEW TREE

The Office Action alleges that Fernandez anticipates:

“generating information to send to said first execution unit to cause said first execution unit to perform said portion..., wherein said information would cause said first execution unit to generate said XML data in a first form that cannot be used by said second execution unit” in column 37, lines 48-61. (Office Action, page 3)

This is incorrect.

In column 37, Fernandez describes generating a View Tree, capturing the nesting relations present in SQL queries and corresponding to the components of the View Tree, but none of the components of the View Tree is actually sent to the first execution unit,” as claimed, and none of the components of the View Tree “causes the first execution unit to generate XML data in a first form,” as claimed. As described above, the View Tree defines a mapping between the XML query entities, relational constructs in the corresponding SQL queries and nested relationship of the constructs, and is useful in combining the SQL results into one XML output document. However, none of the components of the View Tree itself is sent to any of the

RDBMS execution units, as claimed, and none of the components of the View tree is actually used to “generate XML data by the first execution unit,” as claimed.

FERNANDEZ DOES NOT DISCLOSE “ANNOTATING SAID INFORMATION...”

The Office Action alleges that Fernandez anticipates:

“**annotating** said information with an annotation that causes XML data generated by said first execution unit to be transformed to a canonical form for use by said second execution unit in performing said another portion of said query execution plan,”

in column 35, ll. 64-67, column 36, ll. 25-35, and column 28, ll. 1-5. (Office Action, page 3) This is incorrect.

In the columns that are alleged to show this limitation, Fernandez describes a View Tree, which, as discussed above, does not contain any component that can be “sent to an execution unit,” as claimed, and does not “cause XML data generated by said first execution unit to be transformed to a... form for use by said second execution unit,” as claimed.

Instead, in these particular columns, Fernandez describes decomposing user’s RXL query and rewriting each nested pattern from the XML query in a canonical form, and generating a View Tree of the query. Fernandez also describes that each node of the View Tree corresponds to an element in one of the construct clauses in the RXL query, and is annotated by a non-recursive datalog query that computes all instances of that node used to generate the output XML document. (Datalog is a query and rule language for deductive databases that syntactically are a subset of Prolog - a computer language.) From the RXL queries, Fernandez derives the multiplicities of the parent/child relations, stores those relationships and uses them once all RXL queries, translated to SQL queries, return SQL results that need to be translated and integrated into and output XML document. (Fernandez: cols. 35-36) However, Fernandez’ annotations are not used to instruct one execution unit to generate XML data in a particular form and to be sent

to a second execution unit. Fernandez' annotations have no influence on how and in what form the data is exchanged between the execution units. The annotation in Fernandez pertains to annotating the View Tree's nodes of a query, but is not an "annotation that causes XML data generated by the first execution unit to be transformed to a canonical form to use by the second execution unit in performing said another portion of the query execution plan," as claimed.

The Office Action states that Fernandez anticipates "said **annotating** causes removal of one or more references to execution unit-specific data that is accessible by the first execution unit but that is not accessible by the second execution unit" in Fernandez: column 28, ll. 1-10; column 6, ll. 61-67; and column 7, ll. 1-19. (The Office Action, page 3) This is incorrect.

Fernandez' annotations pertain to annotating the View Tree's nodes, which are never sent to the execution units, and thus, Fernandez' annotating cannot "cause removal of one or more references to execution unit-specific data that is accessible by the first execution unit but that is not accessible by the second execution unit," as claimed.

Thus, Claim 1 recites at least one feature that is not anticipated by Fernandez. Therefore, Fernandez does not anticipate Claim 1. Reconsideration and withdrawal of the rejection is respectfully requested.

CLAIM 39

Claim 39 recites:

39. A method for processing XML data, comprising the computer-implemented steps of: receiving information at a first execution unit to cause said first execution unit to perform work associated with a query execution plan for servicing a request for data; wherein said information comprises an annotation that causes the XML data generated by said first execution unit to be transformed to a canonical form for use by a second execution unit; wherein said information, without said annotation, would cause said second execution unit to receive from said first execution unit XML data in a first form that cannot be used by said second execution unit;

transforming XML data generated by said first execution unit to said canonical form prior to providing said XML data to said second execution unit,
wherein transforming XML data comprises removing one or more references to execution unit-specific data that is accessible to the first execution unit but that is not accessible to the second execution unit; and
providing XML data that is transformed to said second execution unit in said canonical form for use in performing work associated with said query execution plan by said second execution unit.

The Office Action states that Fernandez anticipates all features of Claim 39 in Fernandez' FIGS. 6-7, column 6, ll. 61-67, column 7, ll. 1-19, column 28, ll. 1-10; column 6, ll. 61-67; column 35, ll. 64-67; and column 36, ll. 25-35. (The Office Action, page 9) This is incorrect.

Claim 39 recites features similar to those in claim 1, but written from the perspective of the first execution unit. In fact, in rejecting Claim 39, the Office Action relies on the same Fernandez' references as in rejecting Claim 1. All those references were discussed for Claim 1, where it was shown that Fernandez does not anticipate Claim 1.

Since Claim 39 recites features similar to those in Claim 1, and inapplicability of all Fernandez' references to § 101(e) rejection of Claim 39 were discussed for Claim 1, Claim 39 is patentable over Fernandez for the same reasons as for Claim 1. Therefore, Fernandez does not anticipate the whole subject matter recited in Claim 39.

Based on the foregoing remarks, reconsideration and withdrawal of the rejection of Claim 39 is respectfully requested.

CLAIM 20

Claim 20 recites features similar to those in Claim 1, except that Claim 20 is written in the form of a computer-readable storage medium.

Therefore, for the same reasons as for Claim 1, Fernandez does not anticipate the features recited in Claim 20. Reconsideration and withdrawal of the rejection of Claim 20 is respectfully requested.